An Evolutionary Approach to Generate Solutions for Conflict Scenarios

Davide Carneiro, Cesar Analide, Paulo Novais, José Neves

Departamento de Informática, Universidade do Minho, Campus de Gualtar, Braga, Portugal
{dcarneiro, analide, pjon, jneves}@di.uminho.pt

Abstract. Conflict resolution is nowadays an important topic. Online Dispute Resolution in particular is nowadays a major research topic, focusing on the development of technology-based tools to assist parties involved in conflict resolution processes. In this paper we present such a tool aimed at the generation of solutions. It is based on Genetic Algorithms that evolve a population of solutions through successive iterations, generating more specialized ones. The result is a tree of solutions that the conflict resolution platform can use to guide the conflict resolution process. This approach is especially suited for parties which have no ability or are unwilling to generate realistic proposals for the resolution of the conflict.

Keywords: Online Dispute Resolution, Genetic Algorithms, Distributive Negotiation

1 Introduction

Given the current state of courts, new approaches on conflict resolution are needed. Specifically, courts are nowadays unable to deal with the amount and characteristics of new disputes. In fact, while in the past conflicts emerged between persons generally in the geographical vicinity of each other, nowadays a conflict may emerge between two persons, regardless their location, and it may even involve software agents. This new modality of contracting, the so-called electronic contracting, is in fact one of the biggest challenges for current legal systems, relying in paper-based courts still shaped after the industrial revolution [1]. In that sense, alternatives to litigation are needed.

The first ones involved parties trying to solve their differences without recourse to litigation, generally with the assistance of a third, neutral party. These processes include negotiation, mediation, arbitration or conciliation, just to name a few [2-4], and are part of the so-called Alternative Dispute Resolution [3]. However, under these traditional approaches stakeholders still have to meet in person. In that sense, Online Dispute Resolution emerged as the use of traditional conflict resolution mechanisms under virtual environments [5]. In fact, the use of technology for conflict resolution may not only be used to bring parties into contact but also to develop high value tools

that can be used for the definition of strategies, for the generation of solutions or even for compiling useful information for the parties [6,7].

In the last years, our research has focused on developing such tools based on Intelligent System techniques, giving birth to the UMCourt conflict resolution platform [8, 9]. Specifically, we have been researching how the coming together of different fields of research can solve concrete problems in an efficient manner [13, 14], giving birth to the so-called Hybrid Intelligent Systems, applied to the domain of The Law in the specific case of this work. In this work we propose a module for generating solutions for a conflict resolution scenario based on Genetic Algorithms (GA) [10, 15]. Our line of attack targets a very specific issue in conflict resolution: the inability or unwillingness of parties to generate solutions. In fact, frequently parties find it difficult to generate realistic proposals, because they are not fully aware of what their chances are or what rules apply. In other cases, parties are simply uncooperative and do not want to bother creating solutions [11]. With this work we complement our conflict resolution platform with the ability to propose fair and realistic solutions for concrete conflict resolution scenarios.

The rest of the paper is organized as follows. In section 2 we provide a definition of the general conflict resolution scenario and how GAs can be modeled to be used as a problem solving methodology. Section 3 details the initialization of the GA while section 4 is devoted to depicting the selection process. The reproduction operators are detailed in section 5 and the termination of the algorithm is described in section 6. Finally, in section 7 we present the concluding remarks of this work.

2 Defining a Conflict Resolution Scenario with Genetic Algorithms

Even from a technological point of view, the problem of generating solutions may be a challenging task, although several different techniques can be used. One of the possible lines of attack is the use of case-based approaches, in an attempt to shape the cognitive models of human experts which rely on experience. However, this approach has some potential limitations. Specifically, it is likely to fail in scenarios in which case-bases with insufficient cases are used. Moreover, bigger case-bases ensure more completeness but generally also result in slower processes. In this paper we propose an approach that is independent of these constraints: the use of Genetic Algorithms to create solutions for conflict resolution scenarios.

Under GA approaches, a solution for a problem is represented by a chromosome. In that sense, given the domain of application of this work, each chromosome represents a solution for a specific conflict resolution problem, generally a distribution of the items being disputed among several parties. The population of chromosomes evolves from generation to generation through the application of genetic operators that act on the distribution, thus changing its fitness. This approach has also a specificity considering fitness. Usually, in a GA problem, fitness is seen as an absolute value. In this context, a solution has several values of fitness, one for each party, i.e., a solution that is good for a given party is most likely not that good for any other given that they tend to have conflicting objectives. As the generations of solutions succeed, there are lines of evolution of chromosomes that tend to be more fit

to a given party. These lines emerge naturally and are called species. A species is thus defined as a group of chromosomes from the same line of evolution whose fitness is better for a specific party. In that sense, each party has a species of solutions. A chromosome may also belong to more than one species if it has satisfactory values of fitness for more than one party. These chromosomes are evidently more attractive since they correspond to solutions that will be more easily accepted by the parties.

A population *P* of size *s* is defined by a set of chromosomes *Ch* (Figure 1), in which each chromosome $Ch_i, i \in \{1,2,\dots,s\}$ represents a possible solution for the problem, i.e., who gets how much of what. Considering a dispute involving *n* parties and *m* issues, a chromosome *Ch* can be represented as an m-by-n matrix (equation 1).

$$Ch = \begin{bmatrix} V_{1,1} & \cdots & V_{1,n} \\ \vdots & \ddots & \vdots \\ V_{m,1} & \cdots & V_{m,n} \end{bmatrix} \qquad (1)$$

$$P = \left[Ch_1 = \begin{bmatrix} V_{1,1} & \cdots & V_{1,n} \\ \vdots & \ddots & \vdots \\ V_{m,1} & \cdots & V_{m,n} \end{bmatrix} \middle| Ch_2 = \begin{bmatrix} V_{1,1} & \cdots & V_{1,n} \\ \vdots & \ddots & \vdots \\ V_{m,1} & \cdots & V_{m,n} \end{bmatrix} \middle| \dots \middle| Ch_s = \begin{bmatrix} V_{1,1} & \cdots & V_{1,n} \\ \vdots & \ddots & \vdots \\ V_{m,1} & \cdots & V_{m,n} \end{bmatrix} \right]$$

Fig. 1. Under this model a population of size *s* is represented as a set of chromosomes with a cardinality of *s*.

Under this representation, the value $V_{m,n}$ of the chromosome *Ch* represents the amount of issue *m* that the party *n* receives in this specific solution. Evidently, the actual content of the chromosome depends on the domain of the dispute. Likewise, domain-dependent rules must be defined that enforce the correctness of the solutions generated. Let us take as example the general model of distributive negotiation. Under this model, there is a set of items that must be distributed by a number of parties. Traditional scenarios include divorces or winding up of companies. Under this model each entry in the matrix contains a value between 0 and 1 (equation 2), and the sum of the values of each line must at all times be 1 (equation 3). The total amount of resources received by party *n*, R_n, is defined as the sum of the values of column *n* (equation 4).

$$V_{m,n} \in A, \quad A = \{x \in \mathbb{R} \mid 0 \leq x \leq 1\} \qquad (2)$$

$$\sum_{i=1}^{n} V_{m,n} = 1, \forall m \in \{1,2,\dots,m\} \qquad (3)$$

$$R_n = \sum_{i=1}^{m} V_{m,n} \qquad (4)$$

In the development of this model we also take into consideration the possible existence of indivisible items, i.e., items that due to its characteristics or due to a decision of the parties cannot be divided (e.g. many parties do not agree on selling the item to split the value). For each indivisible item *m*, equation 5 applies.

$$V_{m,i} = 1 \Rightarrow V_{m,x} = 0, \forall x \in \{1,2,\dots,n\}, x \neq i \qquad (5)$$

More specific domains may require the definition of additional rules. This allows this model to be applied to virtually every legal domain. Let us take as example the labor law domain. Under this domain, the items in dispute may be of very different nature, ranging from monetary compensations to the possibility or not of being fired. Considering the Portuguese context, a worker being fired without a just cause is entitled to a compensation that ranges from 15 to 45 days of wage for each year of antiquity. This is generally one of the items being distributed (e.g. in a scenario in which the employee receives 30 days of wage for each year of antiquity, the other 15 days are received by the employer). Other issues generally include night or extra hours, the existence or not of complaints against the employee, among others. All these issues may be modeled in this generic model.

In a general way, the GA lifecycle follows the model depicted in Figure 2: it starts with an initialization of a population, usually in a random way. Afterwards, a cycle repeats until an ending condition is met: the fitness of the population is evaluated and then the population evolves through the application of genetic operators that create new populations with different characteristics. In each iteration, the fittest of the population are selected, thus evolving the population. This is the general model of the algorithm presented in this paper, detailed in the following sections.

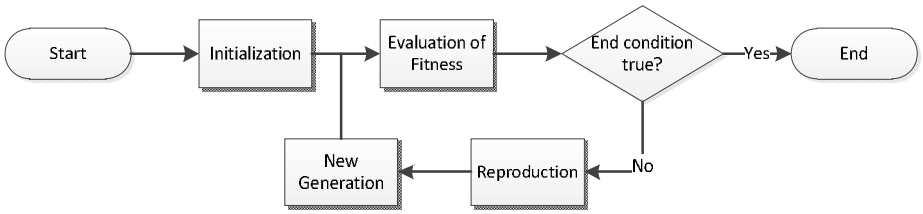

Fig. 2. Generic model of a Genetic Algorithm.

3 Initialization

The initialization is the first step in the use of a Genetic Algorithm, in which the key characteristics of the population and of the individuals are defined. Figure 3 depicts the interface that allows the initialization of the algorithm. In terms of the GA, it is necessary to provide a termination condition, in terms of a maximum number of rounds, and the size of the population (i.e. the number of chromosomes in each generation). The interface also allows to define the number of the best individuals that are selected from each species to create the next generation of offspring through the application of the genetic operators. Concerning these operators, it is possible to specify the weight of each operator on the generation of new solutions. This has, evidently, a significant impact on the evolution of the population. Finally, the interface also allows configuring the weight of the components of the fitness function. Specifically, it is possible to assign the weight of the monetary value and of the personal value. In fact, the measure of the fitness of a solution is given in terms of the monetary value of the items in dispute but also in terms of the personal value, i.e., it is also taken into consideration how much each party wants a given issue.

Concerning the specific information about the negotiation process itself, it is possible to state which are the items under negotiation (including their name, value or type) and which parties are involved. Moreover, each party must assign its preferences regarding the items in dispute. They do so by distributing a total of 100 points for all the items. This information allows the system to know how much each party wants each item and enables an estimation of the personal evaluation of the solutions.

All this information is used to initialize the algorithm. At this point, a population of the specified size is created with random solutions, i.e., each chromosome has a random distribution of items.

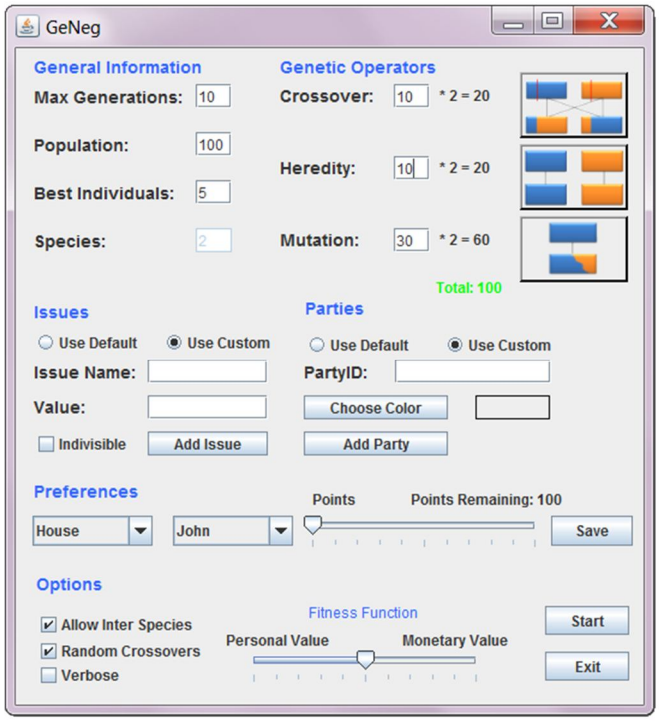

Fig. 3. The system interface used to configure the genetic algorithm, including information about the parties, the issues and the weight of each genetic operator.

4 Selection

In each iteration of the algorithm a part of the population is selected to generate the following generation. This step relies on a fitness function that evaluates each individual and allows finding the best solutions. Given that solutions have different fitness values for different parties, the fitness of each individual must be computed for each party. Thus being, for a conflict resolution involving *n* parties and for a

population of size *s*, *n* * *s* values of fitness will be computed in each iteration of the algorithm.

The fitness function returns a value that is based on the portion of the items that each party receives, its monetary value and the assigned personal value. Moreover, the value of fitness also depends of the weights of the monetary and personal components, defined in the initialization. Two fitness functions were considered in this experiment (equations 6 and 7), where

- *tmv* denotes the case economic value, i.e., the total amount of money that the issues in dispute are worth with, being defined as $tmv = \sum_{i=1}^{I} mv_i$;
- *I* defines the number of issues;
- mv_i stands for the monetary value of issue *i*;
- $fit_{j,p}$ represents the fitness of chromosome *j* for party *p*;
- W_m denotes the weight of the monetary component while W_p stands fot the weight of the individual component.

When equation 6 is used as the fitness function, the solutions selected tend to be the ones in which the parties receive approximately the items that they want. That is, equation 6 minimizes the difference between the personal value of the items and the points attributed to them by the parties.

$$fit_{j,p} = W_m * \frac{\sum_{i=1}^{I} Ch_{j,p} * mv_i}{tmv} + W_p * \left(1 - \sum_{i=1}^{I} \frac{|Ch_{j,p} - prefs_i|}{I}\right) \quad (6)$$

On the other hand, equation 7 tends to result in solutions in which both the monetary and the personal values are maximized. In that sense, solutions selected by the fitness function depicted in equation 6 may more likely be described as fair (as there is no blind maximization of the individual gains) while the ones by equation 7 will be more competitive and hard to be accepted by the opposing parties.

$$fit_{j,p} = W_m * \frac{\sum_{i=1}^{I} Ch_{j,p} * mv_i}{tmv} + W_p * \sum_{i=1}^{I} \frac{|Ch_{j,p} - prefs_i|}{I} \quad (7)$$

Given this, we are currently making use of equation 6 as it results in solutions that are more balanced and thus more likely to be accepted by all the parties. Given this, in each iteration of the algorithm the fittest solutions of each species are selected to give birth to an offspring by means of genetic operators, as depicted in section 5.

5 Reproduction

The reproduction is the step in a GA in which the search heuristic moves forwards, through the engendering of new populations, towards the maximization of the fitness

function. In this work, three genetic operators are being used: crossover, mutation and heredity. All of the three act on the distribution of the items, thus changing its fitness. They are applied to the selected chromosomes according to what was specified during the initialization. The operators used are defined in the following three sub-sections.

5.1 Mutation

A mutation is formally defined in genetics as a spontaneous and random change in a genomic sequence. Transposing this definition for the domain of our work, we can define mutation as a random change in the distribution of the items. The extent of the mutation is given by the mutation threshold, here designated as μ. The mutation is a unary operator that works by randomly selecting one issue and two parties from the chromosome. The distribution is then changed for the item and the parties selected according to μ. If the item is divisible, the amount of the selected item is decreased for one party and accordingly increased for the other, according to μ. On the other hand, if the item is indivisible, there is a probability given in function of μ that the owner of the item is changed between the two parties.

Whenever a new chromosome is created, its validity is checked to determine if all the invariants hold, according to rules of the type of the ones defined in section 2. Let us now consider an example scenario in which three parties are disputing four issues. Let us also assume that issue 2 is divisible and it was randomly selected to be exchanged between party 1 and party 2. The parent chromosome (*Ch*) and the offspring (*Ch'*) are depicted in equation 8.

$$Ch = \begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,3} \\ V_{2,1} & V_{2,2} & V_{2,3} \\ V_{3,1} & V_{3,2} & V_{3,3} \\ V_{4,1} & V_{3,2} & V_{4,3} \end{bmatrix} \quad Ch' = \begin{bmatrix} V_{1,1} & V_{1,2} & V_{1,3} \\ V_{2,1} + \mu & V_{2,2} - \mu & V_{2,3} \\ V_{3,1} & V_{3,2} & V_{3,3} \\ V_{4,1} & V_{3,2} & V_{4,3} \end{bmatrix} \qquad (8)$$

After the application of the mutation operator the fitness of the solution for each party changes. That is, the new solution will most likely be more favorable to party 1 and less favorable to party 2.

Description of the Mutation algorithm.

```
Algorithm Mutation is
Input: List of parties, L
       List of issues, I
       Parent chromosome, C
Output: A new chromosome, C'
Do
   i  := select random issue from I
   p1 := select random party from L
   p2 := select random party from L such that p1 != p2
```

```
    C′ := C
    if (i is divisible)
       C′i,p1 := C′i,p1 + μ * C′i,p1
       C′i,p2 := C′i,p2 - μ * C′i,p2
    else if (randomNumber > μ)
            temp := C′i,p1
            C′i,p1 := C′i,p2
            C′i,p2 := temp
While (C′ is invalid solution)
Return C′
```

5.2 Crossover

In genetics, crossover is a process by means of which a new chromosome is created using the genetic information of more than one parent solutions. In this work, crossover is a binary operator. More specifically, a two-point crossover technique is used. In this specific technique, two points are selected in the two chromosomes and all the information between those two points is swapped. In this precise context, the two points are always the beginning and the end of an issue in the matrix of distribution. Thus being, crossover consists in swapping two distributions of the same issue, generating two new solutions.

Two different approaches can be selected in the initialization form that influence the way that the crossover operator is implemented: *inter species* and *random parents*. The *inter species* option allows the system to cross chromosomes of different species. This will increase the variety of the following generation, but will most likely also delay a convergence. On the other hand, if the *inter species* option is not used, only chromosomes from the same species will be crossed. The *random parents* option tells the system about which parents to cross. If the option is used, parents are selected randomly. On the other hand, if the option is not used, the best parents from each generation are crossed. While the use of this option may increase the variety and widen the search space, it may also delay the convergence towards satisfactory solutions. In equation 9 we depict an example of the use of the crossover operator in two parent chromosomes *Ch1* and *Ch2*, to generate two offspring *Ch1'* and *Ch2'*. In this example the distribution of issue 2 was randomly selected to be swapped. Given that this technique changes the distribution of each solution, it will have effect on the fitness function.

$$Ch1 = \begin{bmatrix} A & B & C \\ D & E & F \\ G & H & I \\ J & K & L \end{bmatrix} Ch2 = \begin{bmatrix} M & N & O \\ P & Q & R \\ S & T & U \\ V & W & X \end{bmatrix} Ch1' = \begin{bmatrix} A & B & C \\ P & Q & R \\ G & H & I \\ J & K & L \end{bmatrix} Ch2' = \begin{bmatrix} M & N & O \\ D & E & F \\ S & T & U \\ V & W & X \end{bmatrix} \quad (9)$$

The description of the generic algorithm that implements the crossover technique being used here.

```
Algorithm Crossover is
Input: List of parties, L
       List of issues, I
Output: New chromosomes, C1', C2'
i  := select random issue from I
if (interspecies)
   s1 = select random species
   s2 = select random species such that s1 != s2
   if (randomparents)
      C1 := select random ch from s1
      C2 := select random ch from s2
   else
      C1 := select best ch from s1
      C2 := select best ch from s2
else
   s1 = select random species
   if (randomparents)
      C1 := select random ch from s1
      C2 := select random ch from s1 such that C1 != C2
   else
      C1 := select best ch from s1
      C2 := select second best ch from s1
swap issues and generate C1', C2'
return C1', C2'
```

5.3 Heredity

Heredity is generally defined as the passing of specific traits from parents to offspring. In this process, the offspring inherits characteristics that may be described as similar to the ones of the parent. During the evolution, the species usually tend to accumulate the best characteristics of their ancestors. In this work, heredity is a very simple unary operator which creates a new chromosome with the same characteristics of the parent, i.e., the same distribution. The objective is to apply this operator to the best individuals only, thus passing the best characteristics of one generation to the next, avoiding losing the best of each generation. However, this operator must be used with caution as an excessive use will result in a population that evolves slowly or that does not evolve at all.

In fact, the weight of each of the above described operators must be chosen appropriately. The *Crossover* operator can be applied thoroughly to the population. However, the *Heredity* and the *Mutation* operators must be applied in smaller amounts. In fact, a big incidence of the *Mutation* operator will significantly increase the variety of the solutions, making it harder for a convergence to emerge. On the other hand, a big incidence of the *Heredity* operator would have the exact opposite

problem, i.e., the evolution would stop and new favorable solutions would hardly appear. In that sense, these two operators can be useful as long as they are used in small proportions.

6 Termination

The process of selection of the fittest and reproduction is repeated until a termination condition is reached: a non-evolving fitness of the population or the number of iterations established in the initialization. At this point, the system has a significant number of solutions. However, some of them will be very similar to each other while others have simply no interest because their value of fitness is low. In that sense, it is not feasible or productive to present the parties or mediators with all this information.

Thus being, only the best solutions of each generation are available to be used by mediators or parties. This helps to simplify the information generated, allowing the users to be focused on what really matters. Figure 4 depicts the simplified view of the information generated, in terms of the solutions attained, and their lines of evolution. Each solution is represented with one or more colors. A solution with a given color means that it belongs to the species of that color. This will allow one to see the natural emergence of species, i.e., the lines of evolution that tend towards the maximization of the fitness value for a given party. Colorless individuals denote solutions that are not among the most fit for a particular population but generate offspring that are among the best of future ones and, for that reason, were included in the group of relevant solutions. It is also possible to look at a chromosome's content, as well to its fitness, and the mean fitness, by clicking on it.

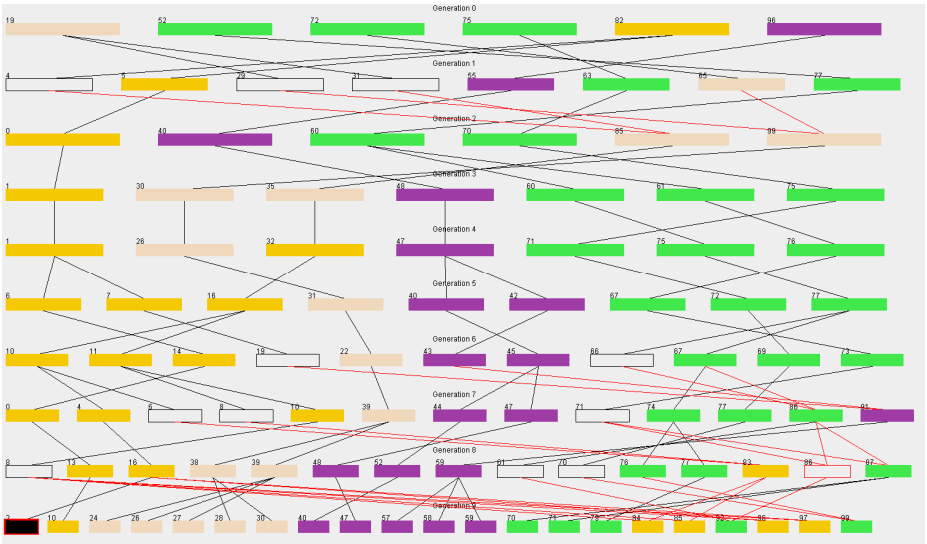

Fig. 4. The lines of evolution of the genetic course and their outcome. Only the individuals that lead to the best leafs are shown.

The lines between individuals represent the parent-offspring relationships. A unary genetic operator generated an individual that has a single line connecting to the previous population, while an individual that has two lines was generated by crossover.

These solutions can then be proposed to the parties by a mediator or by the conflict resolution system. We are currently working on the development of an intelligent conflict resolution environment that is able to collect important information from the context of interaction of the parties [12]. This information includes the levels of stress, the emotional state or the levels of escalation. Based on this, the conflict resolution system or the mediator will select in each time, the most indicated solution for the parties. This will result in a dynamic conflict resolution environment that allows strategies to be adapted in real-time, according to relevant changes in the context-of-interaction.

7 Conclusions

One of the most serious limitations in a conflict resolution process is the inability or unwillingness of parties to design solutions for the resolution of the conflicts. The work described in this paper was developed with the objective of empowering parties and mediators in a conflict resolution process with a tool that is able to provide solutions for concrete problems. Moreover, the solutions generated may be described as fair since they take into consideration not only the monetary value of the items assigned to each party but also the personal value that each party allocates to each item. In that sense, the solutions proposed are more likely to be accepted by the parties.

Compared with our previous case-based approach, this line of attack has as main advantage the independence of a case-base, i.e., the amount and quality of the solutions retrieved does not depend on the quality, quantity or legal domain of the cases in a case-base. In that sense, it provides a more complete answer to the problem. Moreover, despite the computational inefficiency that is generally associated to evolutionary approaches, the performance is good enough for the domain of conflict resolution. In fact, the solutions may be generated as soon as the parties finish providing the data for their case and even before the actual conflict resolution process starts (which is not immediately). In that sense, we can use relatively large parameters on the GA algorithm (e.g. population size, number of generations) ensuring that a big enough number of solutions are generated from which to choose from.

We are now merging this tool into our conflict resolution platform as a solution generation module, to propose solutions during a negotiation process.

Acknowledgments. The work described in this paper is included in TIARAC - *Telematics and Artificial Intelligence in Alternative Conflict Resolution Project* (PTDC/JUR/71354/2006), which is a research project supported by FCT (Science & Technology Foundation), Portugal. The work of Davide Carneiro is also supported by a doctoral grant by FCT (SFRH/BD/64890/2009).

References

1. Katsh, E., Rifkin, J., & Gaitenby, A.: E-Commerce, E-Disputes, and E-Dispute Resolution: In the Shadow of eBay Law. Ohio State Journal on Dispute Resolution, 15, 705 (1999)
2. Raiffa, H.: The Art and Science of Negotiation. Harvard University Press (2002)
3. Brown, H., Marriott, A.: ADR Principles and Practice. Sweet and Maxwell (1999)
4. Bennett, S. C.: Arbitration: essential concepts. ALM Publishing (2002)
5. Katsch E., Rifkin J._ Online dispute resolution – resolving conflicts in cyberspace. Jossey-Bass Wiley Company, San Francisco (2001)
6. Lodder, A., Thiessen, E.: The role of artificial intelligence in online dispute resolution. In Workshop on Online Dispute Resolution at the International Conference on Artificial Intelligence and Law, Edinburgh, UK (2003)
7. Peruginelli, G.: Artificial Intelligence in Alternative Dispute Resolution. In Sartor, G. (Eds.) Proceedings of the workshop on the Law of Electronic Agents (LEA02) (2002)
8. Carneiro, D., Novais, P., Andrade, F., Neves, J.: Retrieving Information in Online Dispute Resolution Platforms: A Hybrid Method. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Law, University of Pittsburgh School of Law, ACM ISBN: 978-1-4503-0755-0 (2011)
9. Carneiro, D., Novais, P., Neves, J.: An Agent-based Architecture for Multifaceted Online Dispute Resolution Tools, in Developing Concepts in Applied Intelligence, Mehrotra K., Mohan C., Oh J., Varshney P., Ali M. (eds), Springer – Studies in Computational Intelligence, ISBN: 978-3-642-21331-1, pp. 89—94 (2011)
10. David E. Goldberg. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Professional, 1st edition (1989)
11. Thomas, K., Kilmann, R.: Conflict and Conflict Management. Available at http://www.kilmann.com/conflict.html (1974) Accessed in: 05/2010
12. Carneiro, D., Gomes, M., Novais, P., Neves, J.: Developing Dynamic Conflict Resolution Models Based on the Interpretation of Personal Conflict Styles. H. Pinto, Antunes L. (Eds.) Progress in Artificial Intelligence, EPIA 2011 - 15th Portuguese Conference on Artificial Intelligence, Springer (2011)
13. Corchado, E., Abraham, A., Carvalho, A.C.P.L.F.D.: Hybrid intelligent algorithms and applications. Inf. Sci. (2010) 2633-2634
14. Corchado, E., Graña, M., Wozniak, M.: New trends and applications on hybrid artificial intelligence systems. Neurocomputing, Vol 75, Issue 1, pp. 61-63 (2012)
15. Holland, J.: Adaptation in Natural and Artificial Systems. Ann Arbor, MI, USA: University of Michigan Press (1975)